18.415/6.854 Advanced Algorithms	November 19, 1992
Lecture 19	
Lecturer: Michel X. Goemans	*Scribe: David B. Wilson*[1]

1 Lattices

Starting with today's lecture, we will look at problems involving lattices and algorithms for basis reduction of lattices. Applications of this topic include factoring polynomials, breaking cryptosystems, rounding an interior point to an optimal vertex in linear programming, and solving integer programs. We start with definitions:

Definition 1 *Given a set of vectors $b_1, \ldots, b_m \in \mathbb{Q}^n$, we define the lattice $L = L(b_1, \ldots, b_m) = \{\sum_{i=1}^m \lambda_i b_i : \lambda_i \in \mathbb{Z}\}$. Thus, L is the set of integral combinations of the vectors b_i.*

Example: $b_1 = (1,2), b_2 = (2,1), n = m = 2$.

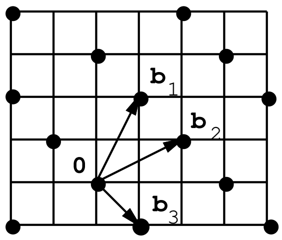

Figure 1: The lattice $L(b_1, b_2) = L(b_2, b_3)$

The simplest lattice is defined by unit vectors; $L(e_1, \ldots, e_n) = \mathbb{Z}^n$.

Definition 2 *A set of vectors $(b_1, \ldots, b_m)$ is a basis for L if $b_1, \ldots, b_m$ are linearly independent (with respect to $\mathbb{Z}$) and $L = L(b_1, \ldots, b_m)$.*

Every lattice has a basis, and its dimension is fixed. A given lattice can have many bases. In the above example for instance, Figure 1 shows that $L(b_1, b_2) = L(b_2, b_3)$. This follows from the fact that $b_3 \in L(b_1, b_2)$ and $b_1 \in L(b_2, b_3)$. The basic operation to obtain another basis for a lattice is to subtract from one of the vectors an integral combination of the others. This idea is presented in our first claim:

Claim 1 $L(b_1, \ldots, b_m) = L(b_1, \ldots, b_{m-1}, b_m - \sum_{i=1}^{m-1} \alpha_i b_i)$ *for $\alpha_i \in \mathbb{Z}$.*

[1]These notes are based on last year's class notes, prepared by Atul Shrivastava and by David Gupta.

Proof: Let $x \in L(b_1, \ldots, b_m)$. Then,

$$x = \sum_{i=1}^{m} \lambda_i b_i = \sum_{i=1}^{m-1} (\lambda_i + \alpha_i \lambda_m) b_i + \lambda_m (b_m - \sum_{i=1}^{m-1} \alpha_i b_i))$$

Since $(\lambda_i + \alpha_i \lambda_m) \in \mathbb{Z}$, we have $x \in L(b_1, \ldots, b_{m-1}, b_m - \sum_{i=1}^{m-1} \alpha_i b_i)$.
Now let $x \in L(b_1, \ldots, b_{m-1}, b_m - \sum_{i=1}^{m-1} \alpha_i b_i)$. Then

$$x = \sum_{i=1}^{m-1} \beta_i b_i + \beta_m (b_m - \sum_{i=1}^{m-1} \alpha_i b_i) = \sum_{i=1}^{m} \lambda_i b_i,$$

where $\lambda_i = (\beta_i - \alpha_i \beta_m)$ for $i = 1, \ldots, m-1$ and $\lambda_m = \beta_m$. □

Definition 3 *L is a full lattice in $\mathbb{Q}^n$ if it can be generated by n linearly independent vectors.*

Example: $L((0,1),(0,3))$ is not a full lattice in $\mathbb{Q}^2$.

Theorem 2 below implies that any one-dimensional lattice has a basis with at most one vector. In the problem set we will show that any lattice in $\mathbb{Q}^n$ has a basis with at most n vectors.

A given lattice can be reduced to a full lattice in polynomial time by restricting our attention to the affine space spanned by the vectors defining the lattice. As a result, without loss of generality, we will look only at lattices that are full.

Also, we will see (as an exercise in the last problem set) that, given a set of vectors $b_1, \ldots, b_m$, a basis for the lattice $L(b_1, \ldots, b_m)$ can be computed in polynomial time. Therefore, without loss of generality, we shall always assume that we are given a full lattice and a basis of that lattice.

Let us show how to compute a basis of L in polynomial time in the case $n = 1$. The general case can be solved in a recursive manner using the result for $n = 1$ (see the problem set). We are thus given m integers $b_1, \ldots, b_m$, and we would like to find an integer a such that $L(b_1, \ldots, b_m) = L(a)$.

Theorem 2 *Let $b_1, \ldots, b_m \in \mathbb{N}$. Then $L(b_1, \ldots, b_m) = L(\gcd(b_1, \ldots, b_m))$.*

Proof: The case $m = 1$ is trivial. Consider the case $m = 2$. Assume w.l.o.g. that $0 \le b_1 \le b_2$. We prove that $L(b_1, b_2) = L(\gcd(b_1, b_2))$ by induction on b_1.

If $b_1 = 0$ then $L(b_1, b_2) = L(b_2) = L(\gcd(b_1, b_2))$. If $b_1 > 0$, then

$$\begin{aligned} L(b_1, b_2) &= L(b_2 - b_1 \lfloor b_2/b_1 \rfloor, b_1) \quad \text{by Theorem 1 above} \\ &= L(\gcd(b_1, b_2 - b_1 \lfloor b_2/b_1 \rfloor)) \quad \text{by the induction hypothesis} \\ &= L(\gcd(b_1, b_2)) \quad \text{by Euclid's algorithm} \end{aligned}$$

The case $m > 2$ is shown by induction on m. Assume the theorem is true for m. Then

$$\begin{aligned} L(b_1, \ldots, b_m, b_{m+1}) &= L(L(b_1, \ldots, b_m), b_{m+1}) \\ &= L(L(\gcd(b_1, \ldots, b_m)), b_{m+1}) \\ &= L(\gcd(b_1, \ldots, b_m), b_{m+1}) \\ &= L(\gcd(b_1, \ldots, b_m, b_{m+1})) \end{aligned}$$

$\square$

Note that the greatest common divisor of two integers can be calculated in polynomial time by Euclid's algorithm, since every two steps reduce the bit size of the maximum by at least 1. By applying Euclid's algorithm repeatedly, we can calculate the GCD of several integers in polynomial time.

2 Combinatorial Application

Suppose we are given a graph $G = (V, E)$, and we want to assign colors to the edges such that no vertex is covered by two edges of the same color, and the number of colors is minimized. The minimum number of colors is at least $d_{\max}$, the maximum degree of any node. Vizing showed that the minimum number of colors is at most $d_{\max}+1$. However, deciding whether the minimum number of colors is $d_{\max}$ or $d_{\max}+1$ is NP-hard, even for special subclasses of graphs.

Consider the class of cubic graphs, the graphs for which every vertex has degree 3. Deciding whether the mininum number of colors needed is $d_{\max} = 3$ or 4 is NP-hard. But if there is a three-coloring, then the edges of the same color make a perfect matching. So there is a three-coloring if and only if there is a partition of E into perfect matchings.

We can identify the perfect matchings M with vectors b in $\mathbb{Z}^{|E|}$. If $e \in M$ then $b_e = 1$, otherwise $b_e = 0$. Let L be the lattice spanned by these vectors. If there is a three-coloring, then $(1, 1, \ldots, 1) \in L$. The converse isn't necessarily true, but this does give us a way to show that a graph is not 3-colorable if we can show that $(1, \ldots, 1)$ is not in the lattice.

3 Shortest Lattice Vector Problem (SLVP)

Given n linearly independent vectors $b_1, \ldots, b_n$ in $\mathbb{Q}^n$ (remember that we can assume w.l.o.g. that we are given a basis of a full lattice), we want to find a nonzero vector $a \in L(b_1, \ldots, b_n)$ such that $\|a\| = \sqrt{a \cdot a}$ is minimized. This problem is called *the shortest lattice vector problem.* Let $\Lambda(L) = \min_{a \in L, a \neq 0} \|a\|$.

The shortest lattice vector problem (SLVP) is believed to be NP-hard. If $\| \ \|$ is replaced by $\| \ \|_\infty$, then it is known to be NP-hard (Van Emde Boas 1981). However, if n is fixed, the SLVP problem is solvable in polynomial time. We will treat below the cases $n = 1$ or 2.

For $n = 1$, the case is trivial since a is a shortest lattice vector in $L(a)$.

Let us now treat the case $n = 2$. We shall find a basis $(b_1, b_2) \in \mathbb{Q}^2 \times \mathbb{Q}^2$ of L in polynomial time in which b_1 is a shortest non-zero lattice vector. We use the 60°-algorithm due to Gauss (1801).

> **If** $\|b_1\| > \|b_2\|$ **then** swap b_1, b_2
> **Repeat** Choose $m \in \mathbb{Z}$ to minimize $\|b_2 - mb_1\|$
> $b_2 := b_2 - mb_1$.
> **Until** $m = 0$
> **Return** b_1, b_2

Claim 3 *The 60°-algorithm terminates in polynomial time.*

The proof is analogous to the proof that Euclid's algorithm terminates in polynomial time (see problem set). As in Euclid's algorithm, the number of iterations is logarithmic in the numbers. The reasons are similar, but more complicated.

Theorem 4 *The 60°-algorithm returns a shortest non-zero vector in L.*

Proof: At termination, we have

- $\|b_1\| \leq \|b_2\|$
- $\|b_2\| \leq \|b_2 - \mu b_1\|$, for all $\mu \in \mathbb{Z}$.

Since $\|b_2\| \leq \|b_2 - \mu b_1\|$ for any integer μ, the orthogonal projection of b_2 on b_1 is between $b_1/2$ and $-b_1/2$ (see Figure 2). On the other hand, $\|b_2\| \geq \|b_1\|$ and so b_2 is outside the circle $(0, \|b_1\|)$. This implies that $|\cos\alpha| \leq 1/2$ and so $60° \leq \alpha \leq 120°$. In fact, because of the hexagonal lattice, this bound is tight.

Let $a = \lambda_1 b_1 + \lambda_2 b_2$ be a shortest non-zero vector in L. Since $\alpha \geq 60°$ and $\alpha + \beta + \gamma = 180°$, we have $\beta \leq \alpha$ or $\gamma \leq \alpha$ (see Figure 3). Therefore we have $\|a\| \geq |\lambda_1|\|b_1\|$ or $\|a\| \geq |\lambda_2|\|b_2\|$, since the length of the sides of a triangle are in the same order as the angles they face. Since the λ_i's are integers and $\|b_1\| \leq \|b_2\|$, this implies that $\|b_1\| \leq \|a\|$. $\square$

Since $60° \leq \alpha \leq 120°$, b_1 and b_2 are almost orthogonal. One can prove that (b_1, b_2) is a couple of independent vectors in L that

1. maximizes $\sin\alpha$
2. minimizes $\|b_1\| \ \|b_2\|$

In fact, we will see that these two statements are equivalent.

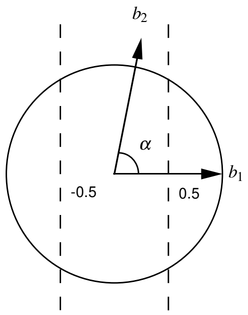

Figure 2: $60° \le \alpha \le 120°$.

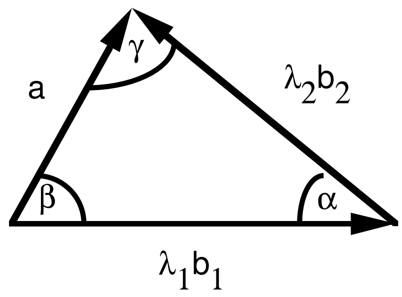

Figure 3: $\beta \le \alpha$ or $\gamma \le \alpha$.

4 Minimum Basis Problem

Given a basis $(b_1, \ldots, b_n)$ of a full lattice $L \subset \mathbb{Q}^n$, consider the non-singular $n \times n$ matrix $B = [b_1 \ldots b_n]$. We know that $|\det(B)|$ is the volume of the parallelepiped defined by $b_1, \ldots, b_n$.

Theorem 5 *Given a full lattice L, $|\det(B)|$ is independent of B, for any basis B of L.*

Proof: Let B and B' be two bases of L. For $1 \le i \le n$, we have $b'_i = \sum_{j=1}^{n} \lambda_{ij} b_j$, where the λ_{ij} are integers. In other words, $B' = BP$, where P is an integral $n \times n$ matrix. Therefore, $|\det B'| = |\det B||\det P|$. But $|\det B'| \ne 0$ since B' is non-singular. Hence $|\det P| \ne 0$ and so $|\det P| \ge 1$ since P is integral. This implies that $|\det B'| \ge |\det B|$. By symmetry, it follows that $|\det B'| = |\det B|$. □

Since $|\det(B)|$ does not depend on the choice of the basis for a given lattice L, let $\det(L) = |\det(B)|$. When $n = 2$, we have $|\det(B)| = \|b_1\| \, \|b_2\| \sin\alpha$, and so minimizing $\|b_1\| \, \|b_2\|$ is equivalent to maximize $\sin\alpha$.

From linear algebra, we know that it is easier to deal with bases which are orthogonal. However, in the case of lattices, this is not always possible. Nevertheless, we are interested in finding a basis that is "somewhat orthogonal". The case for $n = 2$ treated above and Theorem 5 therefore motivates the following problem, called the *minimum basis problem*.

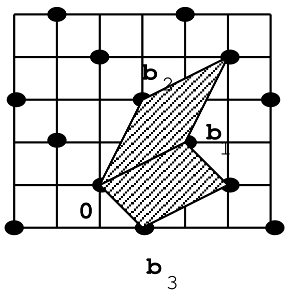

Figure 4: The determinant of a basis is constant in absolute value.

> Given a lattice L, we want to find a basis $(b_1, \ldots, b_n)$ that minimizes the product $\|b_1\| \cdots \|b_n\|$.

This problem turns out to be NP-hard (Lovász). However, there are α- approximation algorithms for this problem where α depends only on the dimension of the basis of the lattice. Fortunately, there is a general lower bound on the size of a minimum basis (which is attained by some lattices and is thus tight) and, in any *given* dimension, *all* lattices have a basis whose size is at most a constant multiple of the general lower bound. This will allow us to develop an approximation algorithm.

Claim 6 *(Hadamard's Inequality). For any basis of L,* $\det L \leq \|b_1\| \ldots \|b_n\|$.

Theorem 7 *(Hermite 1850) For any dimension n, there is a constant c_n such that for any lattice $L \in \mathbb{Q}^n$ there is a basis $b_1, \ldots, b_n$ of L such that $\|b_1\| \ldots \|b_n\| \leq c_n \det L$.*

We will, in fact, take c_n to be the smallest such constant. So what is c_n? In one dimension, a given lattice has only two bases, which are the two minimum nonzero vectors. These vectors are both exactly the size of the lattice spacing, which is also the determinant of the lattice. Then in one dimension, a minimum basis always has a ratio of exactly 1, so $c_1 = 1$. In two dimensions, we know from the analysis of Gauss's algorithm that the angle, α, between the vectors in a minimum basis is at least $60°$. But for basis vectors b_1, b_2, we know that $\det L = \|b_1\|\|b_2\| \sin\alpha \geq \|b_1\|\|b_2\| \sin 60° = \|b_1\|\|b_2\|\frac{\sqrt{3}}{2}$. So $c_2 = \frac{2}{\sqrt{3}}$.

In 1850, Hermite proved that c_n could be bounded by $2^{O(n^2)}$.

In 1984, Schnorr proved that c_n was bounded by n^n.

Unfortunately, neither of the above proofs is algorithmic because neither one gives us any insight on how to actually go about computing a small basis.

In 1983, however, Lenstra, Lenstra, and Lovász provided an algorithm that produces a reduced basis whose size is at most $2^{O(n^2)} \det L$. This algorithm can also be used to approximate the shortest lattice vector problem.

5 More on the Shortest Lattice Vector Problem

Definition 4 *A body, K, is said to be symmetric with respect to the origin if $x \in K \Rightarrow -x \in K$. Note that this statement is its own inverse, so we can think of K being symmetric with respect to the origin as meaning $x \in K \Leftrightarrow -x \in K$.*

We present Minkowski's Theorem without proof as background for a useful corollary.

Theorem 8 *(Minkowski's Theorem-1891) Let K be a convex body symmetric with respect to the origin and let lattice $L \in \mathbb{Q}^n$ be such that $Vol(K) \geq 2^n \det L$. Then K contains a nonzero lattice point.*

Corollary 9 *Consider the norm $\|\cdot\|_p$ for any integer p. Then there is a nonzero $a \in L$ such that $\|a\|_p \leq 2(\frac{\det L}{v_p})^{1/n}$ where $v_p = Vol(\{x : \|x\|_p \leq 1\})$.*

Example: $p = \infty$; $v_\infty = 2^n$. Then there is a nonzero $a \in L$ such that $\|a\|_\infty \leq 2(\frac{\det L}{2^n})^{1/n}$. Thus $(\max_j |a_j|)^n \leq \det L$.

We can give a proof by picture for Corollary 9 when $p = \infty$. Let

$$t = \min_{\substack{a \neq 0 \\ a \in L}} (\max_j |a_j|),$$

i.e. t is the smallest nonzero $\| \ \|_\infty$ norm of lattice vectors. We place a cube with edge length t centered on each lattice point. The cubes may touch, but they don't overlap. The volume per lattice element is t^n. On the other hand, we can cover the space with parallelepipeds of volume $\det L$, one per lattice point. Thus we get the inequality $t^n \leq \det L$.

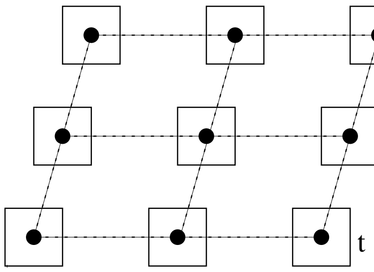

Figure 5: Covering the lattice with cubes and parallelepipeds.

Next time we will discuss L^3, the Lenstra-Lenstra-Lovász theorem. We will give the algorithm and applications.

18.415/6.854 Advanced Algorithms	November 24, 1992
Lecture 20	
Lecturer: Michel X. Goemans	*Scribe: Ray Sidney*[1]

1 Gram-Schmidt Decomposition

First, recall 3 facts about full lattices L and their bases from last lecture:

1. Hadamard's Inequality: If $b_1, \ldots, b_n$ is a basis for L, then $|detL| \leq ||b_1|| \cdots ||b_n||$.
2. Hermite's Thm: $\forall n > 0$, there is a constant c_n such that for any lattice $L \subset \mathbb{Q}^n$, there is a basis $b_1, \ldots, b_n$ of L such that $||b_1|| \cdots ||b_n|| \leq c_n detL$.
3. Corollary to Minkowski's Theorem: For any lattice $L \subset \mathbb{Q}^n$, there is a nonzero vector $a \in L$ such that $||a||_\infty \leq (detL)^{1/n}$ (and hence, such that $||a||_2 \leq \sqrt{n}(detL)^{1/n}$).

We remark that the corollary to Minkowski's Theorem is as good as one can achieve, asymptotically, in the sense that for any $n > 0$, there is a lattice $L \subset \mathbb{Q}^n$ such that $\Lambda(L) \geq \sqrt{\frac{1}{e\pi}}\sqrt{n}(detL)^{1/n}$.

Now remember the Gram-Schmidt Decomposition:

Given the vectors $b_1, \ldots, b_m$, the following procedure calculates m *orthogonal* vectors $b_1^*, \ldots, b_m^*$ such that b_i^* is the projection of b_i onto the space orthogonal to $b_1, \ldots, b_{i-1}$:

$$\begin{aligned} b_1^* &= b_1 \\ b_i^* &= b_i - \sum_{j=1}^{i-1} \frac{(b_i, b_j^*)}{(b_j^*, b_j^*)} b_j^* \end{aligned}$$

The vectors b_i^* are not necessarily in the lattice because the coefficients $\mu_{ij} = (b_i, b_j^*)/(b_j^*, b_j^*)$ are not necessarily integral. We can write $b_i = \sum_{j=1}^{i} \mu_{ij} b_j^*$ where $\mu_{ii} = 1$. Equivalently, $B = B^* P$ where

$$P = \begin{pmatrix} 1 & \mu_{21} & \cdots & \mu_{n,1} \\ 0 & 1 & \ddots & \mu_{n,2} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & 1 \end{pmatrix}$$

Notice that $det(B) = det(B^*)det(P) = det(B^*)$, since all lower triangular entries of P are zero.

Claim 1 $\Lambda(L) \geq \min_i ||b_i^*||$ *for any basis* $(b_1, \ldots, b_n)$ *of* L.

[1] These notes are based largely on notes from Atul Shrivastava, David Gupta, Marcos Kiwi, Andrew Sutherland, and Ethan Wolf.

Most of the proofs of this lecture were omitted in class, but are included for completeness.

Proof: Let $a \in L$ be a minimum-length lattice vector: $\|a\|_2 = \Lambda(L)$. Since $a \in L$, then we can write a as $\sum_{i=1}^{n} \lambda_i b_i, \lambda_i \in \mathbb{Z}$. Let k be the last index for which $\lambda_k \neq 0$. Then $\lambda_j = 0$ for all $j > k$. By substituting in from Gram-Schmidt orthogonalization, we get

$$a = \sum_{i=1}^{n} \lambda_i \sum_{j=1}^{i} \mu_{i,j} b_j^* = \sum_{j=1}^{n} \sum_{i=j}^{n} \lambda_i \mu_{i,j} b_j^*.$$

Let us define λ_j^* for $1 \leq j \leq n$ by $\lambda_j^* \equiv \sum_{i=j}^{n} \lambda_i \mu_{i,j}$. Then $a = \sum_{j=1}^{n} \lambda_j^* b_j^*$. Since the b_j^*'s are orthogonal to each other, we have that

$$\|a\|_2^2 = \sum_{j=1}^{n} (\lambda_j^*)^2 \|b_j^*\|^2 \geq (\lambda_k^*)^2 \|b_k^*\|^2.$$

Thus $\|a\|_2 \geq |\lambda_k^*| \|b_k^*\|$. Note that $\lambda_k^* = \lambda_k \mu_{k,k} + \lambda_{k+1} \mu_{k+1,k} + \ldots = \lambda_k$. Thus $\|a\|_2 \geq |\lambda_k| \|b_k^*\|$. Since $\lambda_k \in Z$ and $\lambda_k \neq 0$, then $|\lambda_k| \geq 1$. So $\|a\|_2 \geq \|b_k^*\| \geq \min_i \|b_i^*\|$. Thus $\Lambda(L) \geq \min_i \|b_i^*\|$. □

2 Lovasz-reduced Bases

In Gauss' algorithm, we were performing swaps to insure that the basis satisfies certain properties. In general, to insure that the first vector of the basis is reasonably short, we shall impose that the switching of any b_i with b_{i+1} does not decrease $\|b_i^*\|$ (recall that the Gram-Schmidt orthogonalization depends upon the ordering of the vectors). This requirement can be more easily stated by using the following observation.

Claim 2 *Let $(b_1, \ldots, b_n)$ be a basis for lattice L. If we switch b_i with b_{i+1} to produce the new basis $(c_1, \ldots, c_n)$, then $b_j^* = c_j^*$ for $j \neq i, i+1$ and $c_i^* = b_{i+1}^* + \mu_{i+1,i} b_i^*$.*

Proof: From Gram-Schmidt orthogonalization, c_j^* is the component of c_j orthogonal to the span of $\{c_1, c_2, \ldots, c_{j-1}\}$, but this set is the same as $\{b_1, b_2, \ldots, b_{j-1}\}$ for $j \neq i+1$. Since $c_i = b_i$ for $j \neq i, i+1$, we get that $b_j^* = c_j^*$ for $j \neq i, i+1$. We also have that c_i^* is the component of $c_i = b_{i+1}$ orthogonal to the span of $\{b_1, b_2, \ldots, b_{i-1}\}$. From the original Gram-Schmidt orthogonalization, we know that $b_{i+1} = \sum_{k=1}^{i+1} \mu_{i+1,k} b_k^*$, so $c_i = \sum_{k=1}^{i-1} \mu_{i+1,k} b_k^* + b_{i+1}^* + \mu_{i+1,i} b_i^*$. Removing the component of each side in the span of $\{b_1, b_2, \ldots, b_{i-1}\}$, we get $c_i^* = b_{i+1}^* + \mu_{i+1,i} b_i^*$. □

This claim says that we can require that switching neighboring basis vectors not help reduce small $\|b_i^*\|$'s by requiring that $\|b_{i+1}^* + \mu_{i+1,i} b_i^*\|^2 \geq \|b_i^*\|^2$ for $1 \leq i < n$.

We would also like for our basis vectors to be as close to orthogonal as possible. If they were strictly orthogonal, then each $\mu_{i,j}$ would be zero. But this is not possible for most lattices. We would like to require that $|\mu_{i,j}|$ be as small as possible for each i, j. We now present a form of these two requirements sufficiently loose enough to guarantee the existence of such a basis and to allow for a polynomial-time algorithm.

Definition 1 *A Lovasz-reduced basis for L is a basis $(b_1, \ldots, b_n)$ for which*

1. $|\mu_{i,j}| \leq \frac{1}{2}$ *for* $1 \leq j < i \leq n$.

2. $\|b^*_{i+1} + \mu_{i+1,i} b^*_i\|^2 \geq \frac{3}{4}\|b^*_i\|^2$ *for* $1 \leq i < n$.

Proposition 3 *Let* $(b_1, \ldots, b_n)$ *be a Lovasz-reduced basis of a lattice* L*. Then*

1. $\|b_1\| \leq 2^{\frac{n-1}{4}} (\det L)^{\frac{1}{n}}$.
2. $\|b_1\| \leq 2^{\frac{n-1}{2}} \min_i \|b^*_i\| \leq 2^{\frac{n-1}{2}} \Lambda(L)$.
3. $\|b_1\| \ldots \|b_n\| \leq 2^{\frac{1}{2}\binom{n}{2}} \det L$.

Proof of Proposition 3:

Claim 4 $\|b_1\|^2 \leq 2^{j-1}\|b^*_j\|^2, 1 \leq j \leq n$.

From property (ii) of a Lovasz-reduced basis and the orthogonality of the b^*_i's, we have

$$\frac{3}{4}\|b^*_i\|^2 \leq \|b^*_{i+1} + \mu_{i+1,i} b^*_i\|^2 = \|b^*_{i+1}\|^2 + \mu^2_{i+1,i}\|b^*_i\|^2.$$

Since from property (i), we know that $\mu^2_{i+1,i} \leq \frac{1}{4}$, we have that

$$\frac{3}{4}\|b^*_i\|^2 \leq \|b^*_{i+1}\|^2 + \frac{1}{4}\|b^*_i\|^2$$

or

$$\|b^*_i\|^2 \leq 2\|b^*_{i+1}\|^2.$$

Repeatedly substituting into the above starting with $i = 1$, we obtain

$$\|b^*_1\|^2 \leq 2^{j-1}\|b^*_j\|^2, 1 \leq j \leq n.$$

Since $b^*_1 = b_1$, this becomes $\|b_1\|^2 \leq 2^{j-1}\|b^*_j\|^2, 1 \leq j \leq n$, which proves the claim. $\diamond$

Solving the above for $\|b^*_j\|^2$, we can square both sides of the definition of $\det L$ and perform a substitution to obtain

$$(\det L)^2 = \prod_{j=1}^{n} \|b^*_j\|^2 \geq \prod_{j=1}^{n} 2^{1-j}\|b_1\|^2 = 2^{\frac{-n(n-1)}{2}}\|b_1\|^{2n}.$$

Raising both sides to the power $\frac{1}{2n}$ gives

$$(\det L)^{\frac{1}{n}} \geq 2^{\frac{-(n-1)}{4}}\|b_1\|$$

$$\|b_1\| \leq 2^{\frac{n-1}{4}}(\det L)^{\frac{1}{n}}.$$

This proves part 1 of the proposition.

Let k be the index for which $\min_i \|b^*_i\|$ is attained, so that $\|b^*_k\| = \min_i \|b^*_i\|$. Then by the above claim: $\|b_1\|^2 \leq 2^{k-1}\|b^*_k\|^2 \leq 2^{n-1}\|b^*_k\|^2 = 2^{n-1}\min_i \|b^*_i\|^2$. Taking the square root of both sides: $\|b_1\| \leq 2^{\frac{n-1}{2}} \min_i \|b^*_i\|$. By applying Claim 1, we can extend this result to: $\|b_1\| \leq 2^{\frac{n-1}{2}}\Lambda(L)$, which is the statement of part 2 of the proposition.

Recall that we have $b_i = \sum_{j=1}^{i} \mu_{i,j} b_j^*$ by Gram-Schmidt orthogonalization. It also follows from the proof of Claim 4, that $\|b_j^*\|^2 \leq 2^{i-j}\|b_i^*\|^2$ for $j < i$. Then making use of the orthogonality and the fact that the coefficients satisfy property (i) of a Lovasz-reduced basis, we get that

$$\|b_i\|^2 = \sum_{j=1}^{i} \mu_{i,j}^2 \|b_j^*\|^2 \leq \|b_i^*\|^2 + \sum_{j=1}^{i-1} \frac{1}{4} 2^{i-j}\|b_i^*\|^2 = \|b_i^*\|^2(1 + \sum_{j=1}^{i-1} \frac{1}{4} 2^{i-j}) \leq 2^{i-1}\|b_i^*\|^2.$$

Multiplying these inequalities for all values of i gives

$$\|b_1\|^2 \ldots \|b_n\|^2 \leq \prod_{i=1}^{n} 2^{i-1}\|b_i^*\|^2 = (\prod_{i=1}^{n} 2^{i-1})(\prod_{i=1}^{n} \|b_i^*\|^2) = 2^{\frac{n(n-1)}{2}}(\det L)^2 = 2^{\binom{n}{2}}(\det L)^2.$$

Taking the square root of both sides gives

$$\|b_1\| \ldots \|b_n\| \leq 2^{\frac{1}{2}\binom{n}{2}} \det L.$$

This proves part 3 of the proposition. □

We now present an algorithm due to A. K. Lenstra, H. W. Lenstra and L. Lovász [2] which computes a reduced basis in polynomial time. We assume throughout that that we are dealing with integral lattices; i.e., we assume that every basis consists of integral vectors.

3 Lenstra-Lenstra-Lovász (L^3) Basis Reduction Algorithm

The algorithm receives as input a set of linearly independent vectors $b_1, b_2, \ldots, b_n \in \mathbb{Z}^n$, and outputs a Lovász-Reduced Basis of $L(b_1, \ldots, b_n)$.

Initialization Find the Gram-Schmidt orthogonalization $(b_1^*, b_2^*, \ldots, b_n^*)$ of $(b_1, b_2, \ldots, b_n)$.

Step 1 Make sure that property 1. of a Lovász-Reduced Basis holds.

> For $i = 2$ to n do
>   For $j = i - 1$ down to 1 do
> $b_i \leftarrow b_i - \lceil \mu_{ij} \rfloor b_j$
>   For $k = 1$ to j do
> $\mu_{ik} \leftarrow \mu_{ik} - \lceil \mu_{ij} \rfloor \mu_{jk}$
>   {Note that for $k > j$, $b_k^* \perp b_j$ so that μ_{ik} is unaffected}

Step 2 If there exists a i for which property 2 of a Lovász-Reduced Basis is not satisfied, swap b_i and b_{i+1}, update the b_k^*'s for $k = i, i+1$, update the μ_{kj}'s for $k = i, i+1$, and go to step 1.

Else return $(b_1, b_2, \ldots, b_n)$.

Note that

1. If the algorithm terminates, it returns a Lovász-Reduced Basis.

2. $b_1^*, b_2^*, \ldots, b_n^*$ are not affected in step 1, since $span\{b_1, b_2, \ldots, b_i\}$, for $i = 1 \ldots n$ is not modified performing this step.

3. After step 1 $|\mu_{ij}| \leq \frac{1}{2}$.

It is not clear that the algorithm makes any progress at each iteration. The following result shows that in fact L^3 terminates.

Theorem 5 *The L^3 algorithm terminates after $O(n^2 \log \beta)$ iterations, where $\beta = \max_i \|b_i^0\|$ (the superscript 0 denotes input vectors).*

Proof: Define a potential

$$\Phi(\bar{b}) = \prod_{j=1}^{n-1} \|b_j^*\|^{2(n-j)} = \prod_{j=1}^{n-1} [\prod_{i=1}^{j} \|b_i^*\|^2] = \prod_{j=1}^{n-1} \det D_j,$$

where $D_j = [d_j(k,l)] = [(b_k, b_l)]_{k,l \leq j}$. Hence, $\Phi(\bar{b})$ is a positive integer, since the D_j's are integral matrices.

In step 1, $\Phi(\bar{b})$ does not change because the b_i^*'s do not change.

In step 2, let $\bar{c} = (c_1, \ldots, c_n) = (b_1, b_2, \ldots, b_{i+1}, b_i, \ldots, b_n)$ be the new basis created after swapping b_i and b_{i+1}. Since c_j^* is the projection of c_j onto the orthogonal of $span\{c_1, c_2, \ldots, c_{j-1}\}$, it follows that $c_j^* = b_j^*$ for $j \neq i, i+1$. Furthermore, $c_i^* = b_{i+1}^* + \mu_{i+1,i} b_i^*$, since c_i^* is the projection of b_{i+1} onto the orthogonal of $span\{b_1, \ldots, b_{i-1}\}$ and b_{i+1}^* is the projection of b_{i+1} onto the orthogonal of $span\{b_1, \ldots, b_i\}$.
Moreover, since $\|b_1^*\| \ldots \|b_n^*\| = \det(L) = \|c_1^*\| \ldots \|c_n^*\|$ we have that $\|b_i^*\| \|b_{i+1}^*\| = \|c_i^*\| \|c_{i+1}^*\|$.

Thus,

$$\frac{\Phi(\bar{c})}{\Phi(\bar{b})} = \frac{\|c_i^*\|^{2(n-i)} \|c_{i+1}^*\|^{2(n-i-1)}}{\|b_i^*\|^{2(n-i)} \|b_{i+1}^*\|^{2(n-i-1)}} = \frac{\|c_i^*\|^2}{\|b_i^*\|^2} < \frac{3}{4}$$

Consequently, the number of iterations of the L^3 algorithm is at most $\frac{\log \Phi_0}{\log 4 - \log 3}$, where Φ_0 is the initial value of Φ.
Now let $\bar{b}^0 = \{b_1^0, b_2^0, \ldots, b_n^0\}$ be the basis given as input. Then $\Phi_0 = \Pi_{j=1}^{n-1} \|(b_j^0)^*\|^{2(n-j)}$. But $\|(b_j^0)^*\| \leq \|b_j^0\|$, thus $\Phi_0 \leq \Pi_{j=1}^{n-1} \|b_j^0\|^{2(n-j)} \leq \beta^{n(n-1)}$, implying that $\log \Phi_0 \leq n(n-1) \log \beta$. It follows that the algorithm terminates after executing step 2 at most $\frac{n(n-1)\log \beta}{\log 4 - \log 3} = O(n^2 \log \beta)$ times. □
(Note that in the proof above the number 3/4 used in condition (ii) could be replaced by $1 - \epsilon$ for any $\epsilon > 0$ and the theorem would still hold).

Corollary 6 *The L^3 algorithm performs $O(n^5 \log \beta)$ arithmetic operations.*

The issue of how large the b_i's can become during the L^3 algorithm was not covered in class. The proof that at any time $size(b_i)$ remains polynomially bounded can be seen in last year lecture notes. In fact it can be shown that at any time $\|b_i\| \leq (1 + 2\beta^{2n+1}\sqrt{n})^n \beta\sqrt{n}$. This result completes the proof that the L^3 algorithm runs in polynomial time.

Beginning this lecture we shall be studying applications of these results in cryptography and simultaneous diophantine approximation. Other applications of the results we have seen relate to polynomial-time integer linear programming for programs of fixed dimension and polynomial-time factorization of polynomials over the rationals.

4 Diophantine Approximation

In a general sense, the Diophantine approximation problem is about how to "round" a number $\alpha \in \mathbb{R}$, meaning that we replace it by a rational number which is of a sufficiently simple form and at the same time sufficiently close to α. If we prescribe the denominator to q of this rational number p/q, then the best choice for p is $\lceil \alpha q \rfloor$. The error resulting from such a rounding is

$$|\alpha - \frac{p}{q}| \leq \frac{1}{2q}.$$

We shall find, however, that often this approximation is not good enough. A classical result of Dirichlet says that if we do not prescribe the denominator, but only an upper bound M for it, then there always exists a rational number p/q such that

$$|\alpha - \frac{p}{q}| \leq \frac{1}{Mq}, \qquad 0 < q \leq M.$$

There also exists a classical method to find such a rational number p/q: this is the so-called *continued fraction expansion* of α. For an irrational number α, this expansion is infinite; for a rational number α, it is finite and of polynomial length.

Khintchine (1956) even showed that continued fractions can be used to solve the following best approximation problem.

> Given $\alpha \in \mathbb{Q}$ (or $\in \mathbb{R}$) and an integer $M > 0$, find a rational p/q with $0 < q \leq M$ such that $|\alpha - p/q|$ is as small as possible.

This often produces very good approximations. For example, if $\alpha = \pi$ and $M = 150$ the best approximation we can obtain using $q \leq 150$ is $355/113 = 3.1415929$.

5 Simultaneous Diophantine Approximation (*SDA*)

Suppose now we wish to approximate several values at once. i.e. we are given $M = 100$ and we wish to approximate $\alpha_1 = .1428, \alpha_2 = .2213, \alpha_3 = .6359$. Note that $\alpha_1 + \alpha_2 + \alpha_3 = 1$. If we approximate each value separately, we find that $\frac{p_1}{q_1} = \frac{1}{7} = .1428\cdots, \frac{p_2}{q_2} = \frac{2}{9} = .2222\cdots, \frac{p_3}{q_3} = \frac{7}{11} = .6363\cdots$. Unfortunately, $\frac{p_1}{q_1} + \frac{p_2}{q_2} + \frac{p_3}{q_3} \neq 1$. Thus, as a group these approximations are not good, since we would like our approximations to mantain "simple" equalities relating the α_i's. This is known as the *SDA* problem and is stated as follows:

> Given $\alpha_1, \cdots, \alpha_n \in \mathbb{Q}$, integer $M > 0$, and $0 < \epsilon < 1$, find $p_1, \cdots, p_n, q \in \mathbb{Z}$ s.t. $0 < q \leq M, |q\alpha_i - p_i| \leq \epsilon$ for all i. (Note that $|q\alpha_i - p_i| \leq \epsilon$ is equivalent to $|\alpha_i - \frac{p_i}{q}| \leq \frac{\epsilon}{q}$).

An equivalent statement of the problem is: given $0 < \epsilon < 1$, $M > 0$, and $\alpha = (\alpha_1, \cdots, \alpha_n)^T$ find $y = (\frac{p_1}{q}, \cdots, \frac{p_n}{q})^T$ such that $\|\alpha - y\|_\infty \le \frac{\epsilon}{q}$. Now, if ϵ is too small, p_i and q may not exist. So we can look at this as a decision problem. Unfortunately, this decision problem has been shown to be NP-complete by Lagarias [1]. It has been shown, however, that for ϵ sufficiently large, a solution always exists.

Theorem 7 *(Dirichlet)* SDA *has a solution if* $M \ge \epsilon^{-n}$.

Proof: Define a lattice $L \subseteq \mathbb{Q}^{n+1}$ by $L = L(b_0, \ldots, b_n)$ where

$$\begin{aligned} b_0 &= (\alpha_1, \cdots, \alpha_n, \delta)^T \\ b_1 &= (-1, 0, \cdots, 0)^T = -e_1^T \\ &\vdots \\ b_i &= (0, \cdots, 0, -1, 0, \cdots, 0)^T = -e_i^T \\ &\vdots \\ b_n &= (0, \cdots, 0, -1, 0)^T = -e_n^T, \end{aligned}$$

where $\delta = \epsilon^{n+1}$. Since $det(L) = \delta = \epsilon^{n+1}$ and $dim(L) = n+1$, by Minkowski's Corollary there exists $a \in L, a \ne 0$ s.t. $\|a\|_\infty \le (det(L))^{\frac{1}{n+1}} = \epsilon$. Hence, there exist $q, p_1, \cdots, p_n \in \mathbb{Z}$ s.t. $a = qb_0 + \sum_{i=1}^n p_i b_i$ with $|a_i| \le \epsilon$ for all i, or, equivalently,

1. $|a_i| = |q\alpha_i - p_i| \le \epsilon$.
2. $a_n = q\delta \le \epsilon$, or, equivalently, $q \le \epsilon^{-n} \le M$.

To complete the proof, we need only check that $q > 0$, (w.l.o.g. we can assume that $q \ge 0$ since we can always take $-a$ instead of a). Now, if $q = 0$ then by 1., $|p_i| \le \epsilon$ for all i. But we know that $p_i \in \mathbb{Z}$ and that $p_i \ne 0$ for some i since $a \ne 0$. However, this contradicts the fact that $0 < \epsilon < 1$. $\square$

Unfortunately, the proof is not constructive, since Minkowski's Corollary insures the existence of a, but doesn't give a procedure for finding it. However, if we make a stronger restriction on the value of M we can find a polynomial time solution to the problem.

5.1 Polynomial Time Algorithm for approximating SDA

We solve the following problem:

> Given $0 < \epsilon < 1, \alpha_1, \cdots, \alpha_n \in \mathbb{Q}$ find $p_1, \cdots, p_n, q \in \mathbb{Z}$ such that $0 < q \le 2^{\frac{n(n+1)}{4}}\epsilon^{-n}$ and $|q\alpha_i - p_i| \le \epsilon$ for all i.

This is a weaker version of the problem, but it can be solved in polynomial time.

To prove this we make use of the L^3 algorithm. But now we use $\delta = 2^{-\frac{n(n+1)}{4}}\epsilon^{n+1}$ in the basis L defined above. Using L^3 we can find $c \in L, c \ne 0$ (the first vector of the

Lovász-Reduced Basis) s.t.

$$\begin{aligned}
\|c\|_\infty &\leq \|c\|_2 \\
&\leq 2^{n/4}(det(L))^{1/(n+1)} \\
&= 2^{n/4}2^{-n/4}\epsilon \\
&= \epsilon
\end{aligned}$$

Hence we can find $q, p_1, \cdots, p_n$ s.t. $c = qb_0 + \sum_{i=1}^n p_i b_i, |c_i| \leq \epsilon_i$ or, equivalently,

1. $|q\alpha_i - p_i| \leq \epsilon$

2. $q\delta \leq \epsilon$ or $q \leq 2^{\frac{n(n+1)}{4}}\epsilon^{-n}$

by solving a simultaneous equation which is done in polynomial time. Note that even though the lattice L is not integral the L^3 algorithm works. Another approach may be to transform the lattice L into an integer lattice before using the L^3 algorithm.

5.2 Maintaining "Simple" Inequalities

We now show that the approximations obtained by this algorithm do in fact maintain "simple" inequalities. Suppose we have an input vector $x \in \mathbb{Q}^n, x = (\alpha_1, \cdots, \alpha_n)$ and we run this vector through the SDA algorithm described above, yielding $y = (\frac{p_1}{q}, \cdots, \frac{p_n}{q})$. Then the following theorem holds:

Theorem 8 *If $ax \leq b$ where $b \in \mathbb{Z}, a \in \mathbb{Z}^n$ and $\sum |a_i| < \frac{1}{\epsilon}$, then $ay \leq b$.*

Proof:

$$\begin{aligned}
b - ay &= (b - ax) + a(x - y) \\
&\geq a(x - y) \qquad \{\text{since } ax \leq b\} \\
&= \sum_i a_i(x_i - y_i) \\
&\geq -(\sum_i |a_i|)\|x - y\|_\infty \\
&> -\frac{1}{\epsilon}\frac{\epsilon}{q} \\
&= -\frac{1}{q}
\end{aligned}$$

But $b - ay$ is rational with denominator equal to q. Therefore, $b - ay \geq 0$. □

5.3 Repairing "Approximate" Inequalities

We saw that a "simple" inequality on x will also hold for its approximation y obtained by our algorithm for simultaneous Diophantine approximation. In fact, if a "simple" inequality "almost" holds for x, then the inequality holds for y, once passed through the SDA algorithm.

Theorem 9 *Let $x \in \mathbb{Q}^n$. Using the SDA algorithm with $\epsilon = \frac{1}{2\beta}$ yields $y = (\frac{p_1}{q}, \cdots, \frac{p_n}{q})$. If $\sum |a_i| < \beta$ for $a_i, b \in \mathbb{Z}$ and if $ax \leq b + (2\beta)^{-n} 2^{-\frac{n(n+1)}{4}-1}$ then $ay \leq b$.*

Proof: The proof is almost identical to the previous proof.

$$\begin{aligned}
ay - b &= (ax - b) + a(y - x) \\
&\leq (2\beta)^{-n} 2^{-\frac{n(n+1)}{4}-1} - \sum_{i=1}^{n} |a_i| ||y - x||_\infty \\
&< \frac{1}{2q} + \beta \frac{\epsilon}{q} \\
&= \frac{1}{2q} + \frac{1}{2q} \text{ since } \epsilon = \frac{1}{2\beta} \\
&= \frac{1}{q}.
\end{aligned}$$

The second inequality follows from the fact that $q \leq \epsilon^{-n} 2^{\frac{n(n+1)}{4}}$ which implies that $(2\beta)^{-n} 2^{-\frac{n(n+1)}{4}-1} \leq \frac{1}{2q}$. Because $ay - b$ is rational with denominator q, this implies that $ay - b \leq 0$. □

5.4 An Example: Linear Programming

We can use the SDA algorithm to find the solution to a Linear Program using the interior point method described in an earlier lecture. After a number of iterations in this method, we will arrive near a vertex which is the optimal solution. We can then use the SDA algorithm to reach the vertex itself.

Consider the following primal and dual linear program:

$$\begin{aligned}
&\min c^T x \\
\text{s.t} \\
&Ax = b \\
&x \geq 0
\end{aligned}$$

and

$$\begin{aligned}
&\max b^T y \\
\text{s.t} \\
&A^T y + s = c \\
&s \geq 0
\end{aligned}$$

We have seen that the interior point method delivers a primal feasible solution and a dual feasible solution for which the duality gap is small (i.e. $2^{-\Theta(L)}$). However, remember that to keep the number of bits polynomially bounded, we need to truncate some of the values. Therefore, the solutions obtained are not exactly feasible, though they are almost feasible. At termination, we have thus x, y, s such that the following conditions hold.

$$\begin{aligned}
Ax &\cong b \\
x &\succeq 0 \\
A^T y + s &\cong c \\
s &\succeq 0 \\
b^T y - c^T x &\cong 0
\end{aligned}$$

Here "$\cong$" means approximately equal to, and "$\succeq$" means greater than or nearly greater than. Using Diophantine rounding, if we choose β appropriately and round the solution then we obtain x', y', s' s.t.

$$\begin{aligned} Ax' &= b \\ x' &\geq 0 \\ A^T y' + s' &= c \\ s' &\geq 0 \\ b^T y' - c^T x' &\leq 0 \end{aligned}$$

as desired. We will not consider how β is chosen here, but this is a technical detail (and, in fact, the analysis we have performed must be refined). By strong duality, x' is an optimal solution for the primal and (y', s') an optimal solution for the dual.

5.5 Breaking Public-Key Cryptosystems

Consider a public-key cryptosystem where the values $a_1, \ldots, a_n \in \mathbb{Z}$ are public key (i.e., are known to all the users of the cryptosystem). To send a message (or cleartext) $e_1, \ldots, e_n \in 0, 1^n$, we compute and transmit the cyphertext $t = \sum_{i=1}^{n} a_i e_i$. Decoding the received transmission, then, requires finding the original $e_1, \ldots, e_n$. Note that, in its full generality (The Subset Problem: Given a finite set $S \subset \mathbb{Z}$ and a number $T \in \mathbb{Z}$, is there a subset of S such that the sum of its members is exactly T?), this decoding problem is NP-complete.

What we want so that we have a useful encryption system here is that:

1. For any cyphertext T, there is at most one $e_1, \ldots, e_n \in \mathbb{Z}$ such that $\sum_{i=1}^{n} a_i e_i = T$.
2. There is some kind of trapdoor information associated with the a_i's which makes rapid decryption possible.

Next lecture, we will look at the basic Merkle-Hellman cryptosystem [3] and see how we can use the L^3 algorithm on it.

References

[1] J. C. Lagarias. Computational complexity of simultaneous diophantine approximation problems. *SIAM Journal of Computing*, 14:196–209, 1985.

[2] A. K. Lenstra, H. Lenstra, and L. Lovasz. Factoring polynomials with rational coefficients. *Math. Ann.*, 261:513–534, 1982.

[3] R. Merkle and M. Hellman. Hiding information and signatures in trap-door knapsacks. *IEEE Trans. Inform. Theory*, IT-24:525–530, 1978.

1 Knapsack Public Key Cryptosystems

We consider the *knapsack public key cryptosystem*. The *public key* consists of n numbers $a_1, \ldots, a_n \in \mathbb{Z}$. Given a message $e_1 \ldots e_n \in \{0,1\}^n$ of n bits, we encode it as t by taking the weighted sum of the bits with respect to the public key:

$$t = \sum_{i=1}^{n} a_i e_i.$$

The message $e_1 \ldots e_n$ is called the *cleartext* and the encoded message t is called the *cyphertext*. The public key satisfies the following properties:

1. Given t, there exists at most one word $e_1 \ldots e_n \in \{0,1\}^n$ such that $t = \sum_{i=1}^{n} a_i e_i$.
2. There exists a *trapdoor* (=some secret information) which allows easy conversion to the e_i's given t.

Example: In the single iteration Merkle-Hellman system [?], we choose a superincreasing sequence of a_i''s. That is

$$a_i' > \sum_{j=1}^{i-1} a_j'.$$

In this case it is extremely easy to solve for the cleartext. The inversion algorithm is:

For $i = n$ down to 1
(Determine e_i)
If $t \geq a_i'$ then $e_i = 1$, $t = t - a_i'$
else $e_i = 0$.

In order to hide the superincreasing sequence, we pick an integer multiplier U and a relatively prime modulus M, such that $M > \sum_{i=1}^{n} a_i'$. We define $a_i = U a_i' \pmod{M}$ to be the public key. The trapdoor for this system is U and M.

[1]These notes are based in part on last year's notes by Ethan Wolf.

If we want to find the cleartext from an encoded $t = \sum_{i=1}^{n} a_i e_i$ and we know the trapdoor, we first compute the inverse U^{-1} of U mod M. Then we derive the decodable cyphertext t' from t and U^{-1} in the following way:

$$t' = tU^{-1} \pmod M = \sum_{i=1}^{n} a'_i e_i \pmod M.$$

The algorithm described above can then be used to invert t'.

For example, let $3, 9, 21, 43$ be the superincreasing sequence of a'_i's. For $U = 33$ and $M = 79$ the sequence of a_i's is $20, 60, 61, 76$. The cyphertext constructed from the cleartext 0110 with respect to the superincreasing sequence of a'_i's is $t' = 30$. The cyphertext with respect to the a_i's is $t = 121$. If we want to find the cleartext from t and we know the trapdoor, we first use Euclid's algorithm to compute the inverse U^{-1} of U mod M, which gives $U^{-1} = 12$. Then, we derive the easily decodable cyphertext t' from t and U^{-1} in the following way:

$$t' = tU^{-1} \pmod M = 121 \times 12 \pmod M = 30.$$

Shamir [?] showed that if the a_i are choosen uniformly from an interval that is too big, this system can be broken in polynomial time using the L^3 basis reduction algorithm for most public keys.

2 Breaking the Knapsack Cryptosystem

Consider the case where the coefficients a_i of the cryptographic system are randomly generated. We show that in this case we can break the cryptosystem with high probability provided that the so-called *density* (ratio of n to the logarithm of the maximum a_i) is low (less than $1/n$).

> Let a_i be independently and uniformly distributed in $[1, 2^{n^2}]$ for $i = 1 \ldots n$. Given $t = \sum_{i=1}^{n} a_i e_i$ for $e_i \in \{0, 1\}$, we will find in polynomial time $e = (e_1, \ldots, e_n)$ with probability approaching 1 as n approaches infinity.

This problem was studied by Lagarias and Odlyzko [?]. We present a version due to Frieze /citeFrieze86. The code breaking algorithm will involve constructing a lattice in which the first basis vector output by the L^3 algorithm will give $e_1, \ldots, e_n$ with high probability. The algorithm described in class must be slightly modified. If

$$t \geq \frac{1}{2} \sum_i a_i \tag{1}$$

then it is exactly as described. Otherwise, we replace t by $\frac{1}{2} \sum_i a_i - t$, apply the algorithm for the previous case and then replace every e_i output by its complement $1 - e_i$. From now on, we assume that (??) holds.

Consider the lattice $L = L(b_0, \ldots, b_n) \subseteq \mathbb{Q}^{n+1}$ where:

$$b_0 = \begin{pmatrix} -Mt \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad \text{and} \quad b_i = \begin{pmatrix} Ma_i \\ 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix}$$

for $i = 1 \ldots n$ and where b_i has $i-1$ 0's before the 1. We take $M > \sqrt{n2^n}$, say $M = 1 + \lfloor \sqrt{n2^n} \rfloor$. The idea is to run the L^3 algorithm on the lattice L and look at the first vector x in the resulting reduced basis. The algorithm for breaking the knapsack public key cryptosystem is a simple two-step procedure:

1. Use L^3 to find a short vector x, which will be the first output vector of L^3.
2. Return $\pm x$.

Before analyzing this algorithm, we make a few observations.

Lemma 1

$$e = \begin{pmatrix} 0 \\ e_1 \\ \vdots \\ e_n \end{pmatrix} \in L.$$

Proof: This lemma follows from the fact that $e = b_0 + \sum_{i=0}^{n} e_i b_i$. □

Also observe that, since $\|e\| \leq \sqrt{n}$, we know that $\Lambda(L) \leq \sqrt{n}$. Therefore,

$$\begin{aligned} \|x\| &\leq 2^{\frac{n}{2}} \Lambda(L) \text{ from the } L^3 \text{ algorithm} \\ &\leq 2^{\frac{n}{2}} \sqrt{n} \\ &< M. \end{aligned}$$

Notice that, iff L^3 returns a basis $(c_0, c_1, \ldots, c_n)$, then if $x = c_0 = \lambda e$ for some integer λ, then $\lambda = \pm 1$. This justifies the second step of the procedure and follows from the properties of bases as shown in the following lemma.

Lemma 2 *If $(c_0, c_1, \ldots, c_n)$ is a basis of a lattice L, and $c_0 = \lambda e$, then $\lambda = \pm 1$.*

Proof: Since $e \in L$, we have $e = \lambda_0 x + \sum_{i=1}^{n} \lambda_i c_i$. So $(1 - \lambda\lambda_0)e = \sum_{i=1}^{n} \lambda_i c_i$. This implies that $1 = \lambda\lambda_0$, since otherwise e, and consequently c_0, would be in the span of $c_1, \ldots, c_n$, which contradicts the fact that $(c_0, c_1, \ldots, c_n)$ is a basis. Now, since λ and λ_0 are integers, it follows that $\lambda = \pm 1$. □

The following theorem demonstrates the effectiveness of this simple procedure for breaking the knapsack public key cryptosystem.

Theorem 3 *If $d \in L$ and $\|d\| < M$, then with probability approaching 1 as n tends to infinity, we have $d = \lambda e$ for some $\lambda \in \mathbb{Z}$. In other words, we show that*

$$Pr[\underbrace{\exists d \in L, \|d\| < M, d \neq \lambda e \text{ for any } \lambda}_{\text{Event } A}] = o(1).$$

Since our algorithm returns x with $\|x\| < M$ this means that with high probability $x = \lambda e$. Moreover, by the above lemma, x must be either e or $-e$.

Proof: Consider a vector d with $\|d\| < M$. If $d \in L$ then it can be expressed as $d = \sum_{i=0}^{n} \lambda_i b_i$. So by the definition of the b_i's we have that

$$d_0 = M\left(\sum_{i=1}^{n} a_i\lambda_i - \lambda_0 t\right), \text{ and } d_i = \lambda_i \text{ for } i = 1, \ldots, n.$$

This equality implies that, if d is in L, d_0 is an integer multiple of M. But since $\|d\| < M$, we have $|d_0| < M$. Hence, $|d_0| = 0$ and consequently

$$\lambda_0 t = \sum_{i=1}^{n} a_i\lambda_i = \sum_{i=1}^{n} a_i d_i. \tag{2}$$

Conversely, if $d_0 = 0$ and $\sum_{i=1}^{n} a_i d_i$ is a multiple of t then d must be in L.

For the sake of the analysis, we define some events. Let event A be defined as in the statement of the theorem. For $d \in \mathbb{Z}^{n+1}$ such that $d_0 = 0$, $\|d\| < M$ and $d \neq \lambda e$ for[2] any $\lambda \in \mathbb{Z}$, define A_d to be the event that d belongs to L. Also, let

$$S = \{d \in \mathbb{Z}^{n+1} : d_0 = 0, \|d\| < M \text{ and } d \neq \lambda e \text{ for any } \lambda \in \mathbb{Z}\}.$$

Then

$$A = \bigcup_{d \in S} A_d.$$

Therefore,

$$\begin{aligned} Pr[A] &\leq \sum_{d \in S} Pr[d \in L] \\ &= \sum_{d \in S} Pr\left[\exists \lambda_0 \in \mathbb{Z} : \lambda_0 t = \sum_{i=1}^{n} a_i d_i\right], \end{aligned}$$

the equality following from the fact that, given that $\|d\| < M$ and $d_0 = 0$, we have that $d \in L$ is equivalent to (**??**).

[2] Remember that e is considered fixed so this statement makes perfect sense.

Now consider a fixed $d \in S$. We first show that we can restrict our attention to a "small" set of possible values for λ_0. Indeed, assuming that $\lambda_0 t = \sum_{i=1}^n a_i d_i$, then

$$\begin{aligned}
|\lambda_0| &= \frac{1}{t}\left|\sum_{i=1}^n a_i d_i\right| \\
&\leq \frac{1}{t}\sum_{i=1}^n a_i |d_i| \\
&< \frac{1}{t} M \sum_{i=1}^n a_i \\
&\leq 2M.
\end{aligned}$$

The strict inequality comes from the fact that $\|d\| < M$ and the last inequality follows from our assumption that t is at least half the sum of the a_i's. Hence, for $d \in S$, we can rewrite

$$Pr[A_d] = Pr\left[\exists \lambda_0 \in \mathbb{Z} : \lambda_0 t = \sum_{i=1}^n a_i d_i\right] = \sum_{|\lambda_0|<2M} Pr\left[\lambda_0 t = \sum_{i=1}^n a_i d_i\right].$$

Fix now λ_0. Since $d \in S$ implies that d is not a multiple of e, there exists an index j such that $d_j \neq \lambda_0 e_j$. Hence,

$$\begin{aligned}
Pr\left[\lambda_0 t = \sum_{i=1}^n a_i d_i\right] &= Pr\left[\lambda_0 \sum_{i=1}^n a_i e_i = \sum_{i=1}^n a_i d_i\right] \\
&= Pr\left[\sum_{i=1}^n (d_i - \lambda_0 e_i) a_i = 0\right] \\
&= Pr\left[a_j = \frac{-\sum_{i\neq j}(d_i - \lambda_0 e_i) a_i}{d_j - \lambda_0 e_j}\right] \\
&\leq \frac{1}{2^{n^2}}
\end{aligned}$$

since the a_i are independently and uniformly distributed over $[1, 2^{n^2}]$. As a result, we obtain that

$$Pr[A_d] < 4M \frac{1}{2^{n^2}}.$$

Summarizing, we now have $Pr[A] \leq |S| \frac{4M}{2^{n^2}}$. It thus remains to get an upper bound on $|S|$. Clearly,

$$S \subseteq \{d \in \mathbb{Z}^{n+1} : d_0 = 0 \text{ and } |d_j| < M \text{ for all } j\}.$$

This implies that $|S| \leq (2M)^n$. Finally, we have:

$$\begin{aligned} Pr[A] &\leq \frac{(2M)^n 4M}{2^{n^2}} \\ &= O\left(\frac{\left(2\sqrt{n2^n}\right)^{n+1}}{2^{n^2}}\right) \\ &= o(1). \end{aligned}$$

And this is what we wanted to prove.

$\square$

3 Approximation Algorithms

For the rest of this class we will be covering approximation algorithms. Many optimization problems are NP-complete, so we are willing to settle for a suboptimal solution which isn't too far from the optimum. We will look at worst-case performance.

Definition 1 *The performance guarantee of a heuristic algorithm for a minimization (maximization) problem is α if the algorithm is guaranteed to deliver a solution whose value is at most (at least) α times the optimal value.*

Definition 2 *An α-approximation algorithm is a polynomial time algorithm with a performance guarantee of α.*

Even though NP-complete problems have equivalent complexity when exact solutions are desired, the reductions don't necessarily preserve approximability. The class of NP-complete problems can be subdivided according to how well a problem can be approximated. Papadimitriou and Yannakakis defined the subclass MAX-SNP (MAXimization, Strict NP) which we will describe below.

Consider for example the problem 3SAT. We are given a set of clauses, where each clause is the disjunction of three literals. (A literal is a variable or its negation.) We want to know if there is a way to set the variables true or false, such that every clause is true: we want to know if

$$\exists T \forall c \exists x (P(x,c) \wedge x \in T) \vee (N(x,c) \wedge x \notin T)$$

(Read "Does there exist a truth assignment T such that for all clauses c there is a variable x such that either x appears positively in the clause and is set true, or x appears negatively in the clause and is not set true.") In general, any NP-predicate can be represented as

$$\exists s \forall c \exists x \phi(s,c,x).$$

Now since in 3SAT each clause has a bounded number of variables, we can write it as

$$\begin{aligned} \exists T \forall (x,y,z) \qquad & [(x,y,z) \in C_0 \Rightarrow x \in T \vee y \in T \vee z \in T] \wedge \\ & [(x,y,z) \in C_1 \Rightarrow x \in T \vee y \in T \vee z \notin T] \wedge \cdots \wedge \\ & [(x,y,z) \in C_7 \Rightarrow x \notin T \vee y \notin T \vee z \notin T], \end{aligned}$$

where we have partitioned the clauses C into $C_0, \ldots, C_7$ according to which literals are negated variables. In general, if an NP predicate can be written as

$$\exists s \forall c \phi(s,c),$$

then it is called an SNP, or strict NP predicate.

Instead of asking that for each c we get $\phi(s,c)$, we can ask that the number of c's for which $\phi(s,c)$ is true be maximized:

$$\max_s \#\{c : \phi(s,c)\} \tag{3}$$

In this way, we can derive an optimization problem from an SNP predicate. These maximization problems comprise the class MAX-SNP.

Example: In MAX 2-SAT we are given clauses each having two literals and a weight, and we want to maximize the sum of the weights of the clauses that are true. If the weights are integral and bounded by a constant, then we can formulate this problem as (**??**) by letting there be w separate c's for each clause of weight w For this problem there is a 3/4-approximation algorithm.

If for two MAX-SNP problems π and π', there is some β such that the existence of an α-approximation algorithm for π' implies the existence of an $\alpha\beta$-approximation algorithm for π, then we say that π has been L-reduced to π'. A MAX-SNP complete problem is a MAX-SNP problem to which any MAX-SNP problem can be L-reduced.

Example: The travelling salesman problem with edge weights chosen from $\{1,2\}$ is a MAX-SNP complete problem. There is a 7/6 approximation algorithm for it.

Consequently, for any MAX-SNP problem, there is an α such that there is an α-approximation algorithm for the problem. This year Arora et al [**?**] showed that for any MAX-SNP complete problem π, there is some ε such that there is no $1+\varepsilon$-approximation algorithm for π, unless P=NP.

References

[1] S. Arora, C. Lund, R. Motwani, S. M., and M. Szegedy. Proof verification and hardness of approximation problems. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 14–23, 1992.

[2] J. Lagarias and A. Odlyzko. Solving low density subset sum problems. In *Proceedings of the 25th Annual IEEE Symposium on Foundations of Computer Science*, 1983.

[3] R. Merkle and M. Hellman. Hiding information and signatures in trap-door knapsacks. *IEEE Trans. Inform. Theory*, IT-24:525–530, 1978.

[4] A. Shamir. A polynomial-time algorithm for breaking the merkle-hellman crypto system. *IEEE Transactions on Information Theory*, 30:699–704, 1982.